Computing and Informatics, Vol. 31, 2012, 1003–1024

FIVO/QSTORMAN SEMANTIC TOOLKIT FOR SUPPORTING DATA-INTENSIVE APPLICATIONS IN DISTRIBUTED ENVIRONMENTS

Renata Słota, Darin Nikolow, Jacek Kitowski

AGH University of Science and Technology
ACC Cyfronet AGH
ul. Nawojki 11, 30-950 Krakow, Poland
&
AGH University of Science and Technology
Faculty of Electrical Engineering, Automatics, Computer Science and Electronics
Department of Computer Science
al. A. Mickiewicza 30, 30-059, Krakow, Poland
e-mail: rena@agh.edu.pl

Dariusz Król, Bartosz Kryza

AGH University of Science and Technology
ACC Cyfronet AGH
ul. Nawojki 11, 30-950 Krakow, Poland

Communicated by Ernest Jamro

Abstract. In this paper we present a semantic-based approach for supporting data-intensive applications in distributed environments. The approach is characterized by usage of explicit definition of non-functional quality parameters regarding storage systems, semantic descriptions of the available storage infrastructre and monitoring data concering the infrastructure workload and users operation, along with an implementation of the approach in the form of a toolkit called FiVO/QStorMan. In particular, we describe semantic descriptions, which are exploited in the storage resource provisioning process. In addition, the paper describes results of the performed experimental evaluation of the toolkit, which confirm the effectiveness of the proposed approach for the storage resource provisioning.

1 INTRODUCTION

Modern scientific research becomes more and more dependent on the access to high performance computational and storage resources, usually available in distributed form through various Grid or Cloud platforms. Over the last 10 years, technology oriented scientific activities on scheduling, load balancing and point-to-point data transfer has been sufficiently advanced to offer higher quality level of research in that domain.

Although the available Grid or Cloud middleware solutions support scientists in accessing to the high performance computing infrastructure, the collaboration between different groups of researchers performed completely on top of the Grid environment, as required by e-Science community, is still very limited. The concept of Virtual Organization (VO) [1], introduced in the early papers on the Grid, is still constrained to basic solutions such as VOMS [2], which have no means of supporting complex collaboration functionality such as definition and enforcement of agreement between the partners, advanced security and access control, monitoring of VO level Quality of Service and VO dynamics concerning participation of partners as well as automatic detection of Service Level Agreement violations. These issues have been discussed in depth in [3], including identification of such challenges as governance of Virtual Organizations, metrics and assessment, standards and infrastructure, policies and contracts as well as automation of the VO deployment process.

In order to address these issues we have developed a framework called FiVO (Framework for Intelligent Virtual Organizations) [4], which aims at providing a comprehensive tool for enabling collaborations in distributed environments through semantic based VO management platform, supporting dynamic VO inception, including such functionality as partner discovery, VO goal definition, contract negotiation and VO execution monitoring and contract enforcement.

However, based on the analysis of VO challenges from [3], another important issue in large scientific collaborations is still missing, i.e., support for SLA controlled and QoS aware data management in data intensive applications [5]. In this paper we present a new component of the FiVO system, called QStorMan, which addresses that problem. QStorMan uses the information stored in the VO contract related to the non-functional requirements of data management, and thus allowing to significantly improve the data management and performance for data-intensive jobs within a Virtual Organization. Due to the fact that QStorMan is a semantic toolkit it is possible to achieve semantic interoperability between the various elements of the FiVO system.

The paper is organized as follows. In Section 2 we present related work on the area of optimization of data management in distributed environments. In Section 3 we present the overview and architecture of our system. In Section 4 we present the semantic approach to description of storage resources and user requirements. In Section 5 we present the experimental evaluation results and finally conclude the paper in Section 6.

2 RELATED WORKS

Although semantic technologies have been existing for several years now, it is hard to find their adoption to the data management problem in distributed environments. The state of the art systems use semantics mainly to integrate information gathered from heterogeneous subsystems or components. Knowledge which is stored in ontologies can be used by various clients as a shared data model with a defined meaning.

Such an approach is described in [6] with regards to Cloud environments. The presented system, called "eCloudManager", exploits semantic technologies to address the topics of data integration, collaborative documentation and annotation and intelligent information access and analytics. The "eCloudManager" suite is a multi-layer, Java-based software solution, which allows to manage a highly heterogeneous and changing set of resources encountered in enterprise data centers. As a result, provisioning time and the complexity of the support environment can be dramatically reduced. However, the "eCloudManager" suite can be used only by administrators of a company resources to increase their efficiency. There is no solution for automating some of the cumbersome work, e.g., resource provisioning or data management. Thus, it can be used only to decrease the size of the management problem rather than eliminating it.

In the data management field, one of the few tools, which uses knowledge is the "integrated Rule Oriented Data System" (iRODS) [7] tool. It is a data management system that organizes distributed data and their metadata in the Grid. It is a successor of Storage Resource Broker (SRB). iRODS uses knowledge in the form of rules instead of ontologies to define policies of managing data. A component which interprets the rules, called "Rule Engine", allows to define data storage, data access and data processing in a flexible way. Altough iRODS exposes a set of interfaces to interact with, there is no means to supporting *legacy code*, i.e., applications, which cannot be modified. The most similar suitable interface for supporting *legacy code* is the iRODS Standard I/O library, which can handle C applications only that will access files with the regular I/O functions but each file name has to be prefixed with "irods": string, which needs source code modification.

Though it is hard to find semantic supported systems for data management, many researchers addressed the issue of data management in the Grid. Most of the existing solutions have covered the problem of reading data rather than storing data. One of the most commonly exploited methods is the replication. In [8], a concept of a self-managing replication system for Digital Libraries is proposed. Such a system will provide transparent and consistent access to distributed data. It dynamically controls the creation and maintenance of replicas. The described system will be based on an accepted replication protocol for database clusters. Unfortunately, only a concept of the system is presented and no implementation is available yet. There is a number of publications regardings different algorithms concerning the replication mechanism like [11, 12, 13, 14]. For example, in [11] the authors describe a modified version of the *Bandwith Hierarchy Replication* algorithm. It is based on the network

level locality of Grid sites. The algorithm tries to replicate popular files within a region where broad bandwidth is provided within sites.

An existing system for transparent access to remote data by using native I/O function calls is described in [9]. The system, called "Spigot", addresses both access transparency and latency hiding. For access transparency, "Spigot" provides a client application with a global namespace to access geographically distributed files. For latency hiding, "Spigot" uses the on-demand scheme for file transfering to avoid unnecessary data transfer and also adopts a co-allocation (parallel) download strategy and a pre-fatching strategy to improve data transfer performance. The system is implemented using the Filesystem in Userspace (FUSE) [10] project. Although, "Spigot" does not impose any changes to applications, it requires modification in a worker node operating system. Specifically, a dedicated kernel module has to be loaded at each worker node. Then, a dedicated directory is created ("/Spitgot") where all the files of an applications must be written to access the "Spigot" functionality. While "Spigot" always finds the most efficient storage node at the time to either store or get data, it does not enable the end users to decide, which parameters of the storage system they are interested in. It rather uses the "one size fits all" rule, which is not true in a general case and can lead to infrastructure throughput reduction.

3 FIVO/QSTORMAN TOOLKIT OVERVIEW

The FiVO/QStorMan toolkit is a set of tools developed to facilitate the data management process based on an explicit definition of non-functional requirements for storage resources. The toolkit is a subset of Framework for Intelligent Virtual Organizations (FiVO), which is a more general framework for managing Virtual Organizations.

The main features of the FiVO system include:

Partner discovery and selection – FiVO supports the idea known from the VO research community called Virtual Breeding Environment (VBE), which defines a group of organizations and entities, which are interested in collaborating on pursuing some goal. FiVO provides means for each entity to publish their semantic description, including capabilities and resources, which can be then discovered by other partners. The VBE is implemented using a distributed semantic registry, where each organization can publish description of their resources in semantic form. In order to facilitate this process, we have developed a tool called X2R [15] for automating the process of translating legacy metadata description from relational, LDAP or XML databases into Web Ontology Language – OWL [16].

Definition of VO goal – in order to properly build and operate a VO a clear goal needs to be defined and analysed from the perspective of its realization, which is also one of the key points in the above mentioned paper [3]. FiVO allows to define the goal of the VO in terms of products and metrics defined semantically,

which can be evaluated at any point during the VO operation and any deviations from the agreement can be identified.

Contract negotiation – this issue is another major aspect of future partners collaborations based on the idea of dynamic Virtual Organization supported by distributed computing platforms such as Grid. FiVO allows the partners of the VO to negotiate a formal agreement describing their cooperation within the VO, their responsibilities and obligations, resources they will provide to the VO, and roles each partner and users will play in the VO as well as security assertions regarding these roles. Furthermore, the contract allows definition of Service Level Agreement including various QoS parameters, between any combination of parties within the VO. The format is defined using Web Ontology Language, which allows extensions to the generic schema by provision of domain specific concepts and relation describing particular application. The entire process is performed in a fully distributed manner and supported by Eclipse [17] based Graphical User Interface for negotiations.

Security configuration – based on the contract, the security enforcement component is able to automatically configure the underlying security infrastructure using the definition of roles, attributes and access control rules from the contract and translated automatically to VOMS [2] or any XACML [18] based authorization system. It identifies the different roles within the contract, their obligations and restrictions, as well as access control rules, and generates access policies, which allow the members and services of the new VO to gain access to necessary resources within the infrastructure.

SLA monitoring – the system provides means for continuous monitoring on VO level of the fullfilment of the VO contract, based on low level monitoring information collected from the monitoring systems already deployed in the infrastructure, thus allowing notification and possible reaction to any irregularities in the VO operation. The reaction service can perform several actions including notification of responsible users or services, automatic deployment of additional services in order to improve certain Quality of Service metric or stopping of the VO due to some major problem.

3.1 The QStorMan Toolkit

While FiVO aims at supporting every part of the VO lifecycle, from the process negotiation to the SLA enforcement, QStorMan focuses on the QoS provisioning for data-intensive applications. As a framework input, a set of non-functional requirements which should be maintained is given. As a result, a location of storage nodes which are suitable for the given requirements is returned. The architecture of FiVO/QStorMan is depicted in Figure 1. The toolkit consists of the following components:

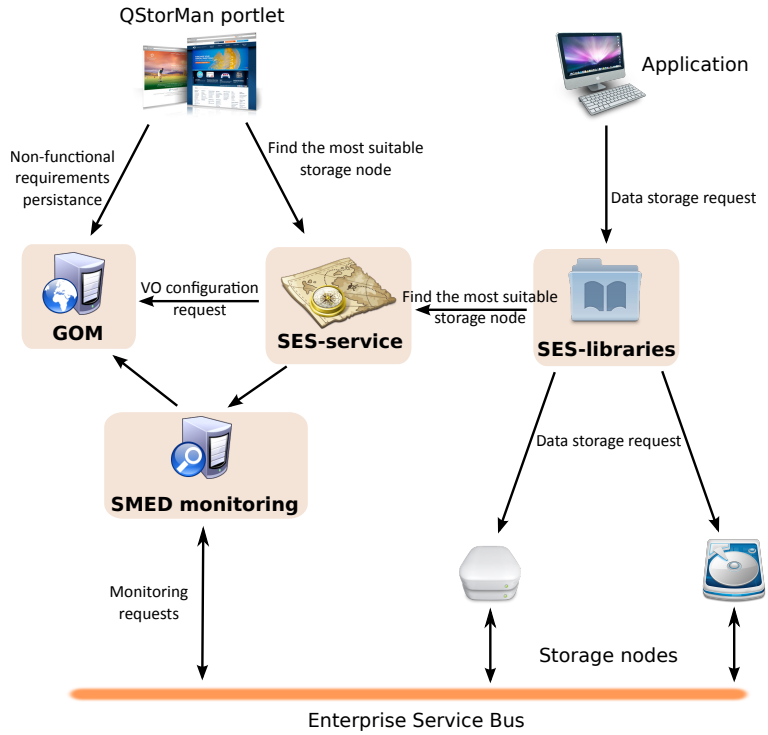


Fig. 1. The architecture of the QStorMan toolkit

- A distributed knowledge based called *GOM* [19], which stores all the information required in order to support inception and management of Virtual Organizations, including semantic description of organizations resources, capabilities and interests, the contracts of all Virtual Organizations as well as domain specific knowledge related to particular applications. Moreover, *GOM* is responsible for storing configuration of the storage environment along with defined non-functional requirements from the users. It exposes an interface for SOAP-based web services.

- A monitoring system – *SMED* – which monitors storage resources and provides information about values of different QoS parameters, e.g., *freeCapacity* or *averageWriteTransferRate*. The *SMED* subsystem exposes an interface compatible with the REST model [20], which supports clients developed with different programming languages as well as using different technologies. The *SMED* subsystem exploits semantic descriptions of different storage resources to monitor their specific features in a plugin-based way, i.e., the core of the *SMED* subsystem is independent of concrete set of storage resource types and can be extended to monitor other types of storage resources easily. The subsystem‘s implementa-

tion is based on the Enterprise Service Bus (ESB), thus it can scale with ease if necessary.

- A portal – *QStorMan portlet* – is a web-based interface where the users can define non-functional requirements for storage resources. In addition, the users can search for a worker node in a distributed environment, which meets the given QoS parameters at the moment. As a result of this action, the *QStorMan portlet* returns a part of a Job Description Language (JDL) [21] file, which can be used to send the Grid job to the selected worker node. The *QStorMan portlet* is based on the open-source Liferay portal [22] and meets the JSR-168 portlet specification [23] in order to be integrable with a wide set of existing portals easily. A screenshot from the *QStorMan portlet* is depicted in Figure 2.

- A service – *Storage Element Selection service* (*SES-service*) – is the central element of the *QStorMan* toolkit. It processes requests for finding storage nodes or worker nodes, which meet QoS requirements defined for one of the following object: an application, user or Virtual Organization. The defined requirements are taken by the *SES-service* from the *GOM* knowledge base. The service exposes an interface compatible with the REST model, which is both language and technology independent.

- On the user side the *QStorMan* toolkit provides two programming libraries for interacting with the server side part of the toolkit, called *SES-libraries*. The first library – called *libSES-wrapper* – is a system-level C library for supporting applications without modifying their source code, which are often called *legacy applications*. The *libSES-wrapper* works at a filesystem access level, allowing to intercept applications file creation requests in which applications data will be stored. Based on the QoS requirements stored in the knowledge base, the files are created on suitable storage nodes. To activate the *libSES-wrapper* library, either the user who starts an application or the worker node administrator has to set the `LD_PRELOAD` system variable to the path to the *libSES-wrapper* library. This technique is known as the library pre-loading technique. The second library – called *libSES* – provides an Application Programming Interface (API) in C++ for managing file creation from the application source code level. It can be used to create new Grid applications, which heavily exploit storage systems. Using this library, application developers can store the applications data with more control. The *libSES* library is a standard C++ dynamic programming library and can be used as any other library of this type.

Most of the described components are located at the server-side in order to decrease the amount of computation on the user(application)-side.

The *QStorMan* toolkit is developed using modern technologies and models like ESB and REST. Also, the scalability of the implementation was particularly taken into account in order to handle a large number of users requests simultaneously.

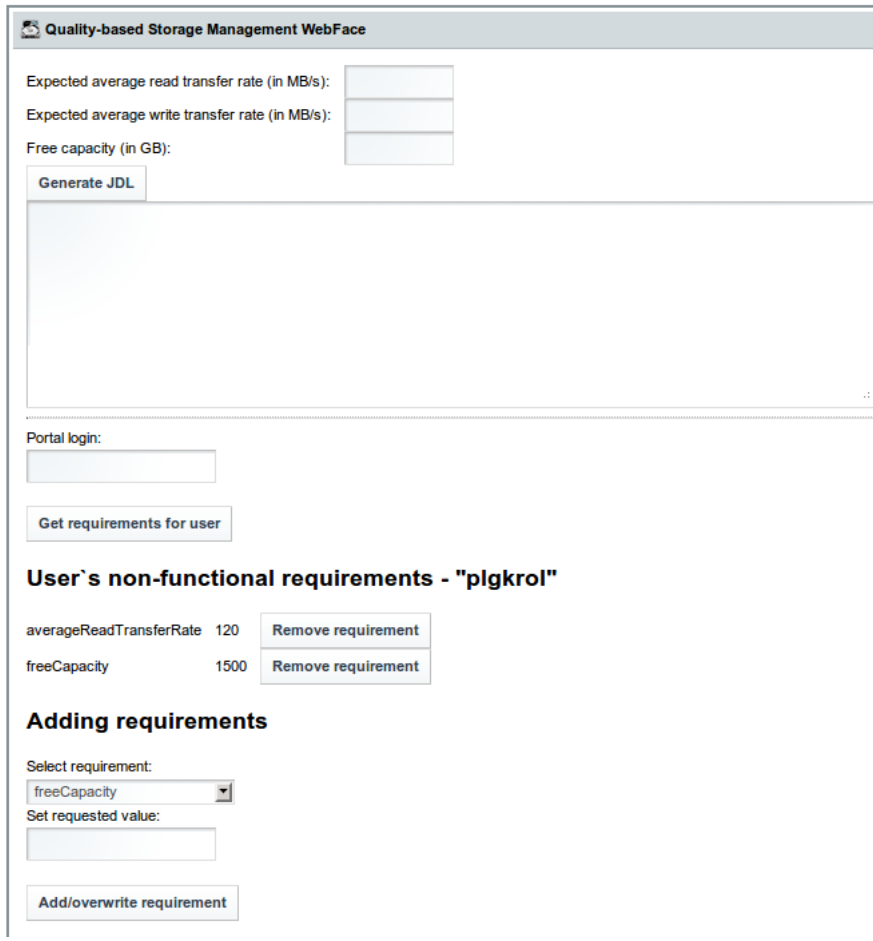


Fig. 2. The *QStorMan* portlet

4 SUPPORTING STORAGE RESOURCES PROVISIONING WITH SEMANTICS

The QStorMan toolkit exploits semantic descriptions to integrate different components with a single, shared data model. By using ontologies stored in the GOM knowledge base, each of the *QStorMan* toolkit subsystems uses the same set of concepts with a defined meaning. In ontologies, the *QStorMan* toolkit stores information about available storage resources and defined non-functional requirements for different subjects, i.e., VOs, users or applications.

Information about available storage resources and non-functional requirements for storage resources can be generated based on a VO contract. In such a case, the

VO contract created during the negotiation phase is represented with statements from the Contract Ontology. Statements regarding storage resources and required QoS are transformed to QStorMan specific ontologies.

In the following subsections, we describe how and which ontologies are used to represent VO contracts and then which information is especially important for the QStorMan toolkit.

4.1 Ontologies in FiVO

VO contract is created using concepts defined within the Contract Ontology in Web Ontology Language. The Contract Ontology is constituted with a set of concepts representing possible contract statements. Each type of statements allows for representation of different contract elements. Main types of statements covered by the Contract Ontology include:

- VO lifetime
- resource provision
- QoS requirement
- role assignment
- role definition
- authorization policy
- security requirement
- penalty clause.

These concepts provide only generic structure of the contract, which can be automatically procesed by the components of the FiVO framework. However, all of them can be extended by definition of domain specific concepts related to the application at hand, which will be still processable by the FiVO tools. The Contract Ontology depends on several ontologies in order to define additional concepts related to the attributes of the statements, such as QoS ontology (QoSOnt2), VO ontology and security ontology [24].

Beside the Contract Ontology, there are two more ontologies exploited by the FiVO system and the QStorMan toolkit in particular. The first one is the Storage Resource Ontology (described in Subsection 4.2), which contains information about the available storage environment. The second one is the Requirement Ontology, which includes statements regarding non-functional requirements for storage systemns. The Requirement Ontology is described in Subsection 4.3 in more detail.

4.2 Storage Resource Ontology

Definitions of different storage resources available in a distributed environment are described in a single ontology called *Storage.owl*. In this ontology, different types

of storage resources were defined and described along with their specific features. The ontology is depicted in Figure 3. It is based on the previously created ontology within the OntoStor project [25] and is compatible with the C2SM model [26]. Each type of storage resource has a number of attributes attached (represented by the *Attribute* class), each representing measurable quality parameter of the resource, e.g., `currentReadTransferRate` or `freeCapacity`. All storage resources have been divided into two groups: *physical* storage resources, which represent hardware devices dedicated to store data, e.g., hard drives or disk arrays. Such devices are often accessible via a standard file system.

However, if there is a need to provide users with a coherent view of very large capacity, one can aggregate several hardware devices (physically distributed) into a single system. Therefore, a concept of a *virtual* storage resource has been introduced. This class encompasses, e.g., distributed file systems, which can consist of several physical devices. An example of such a resource is a concept of a *pool* in the Lustre file system [27], which simply describes a set of disks.

Each storage resource needs an access server through which the users can access provided storage space. In our ontology, this type of service is represented by the *AccessResource* concept, which is linked with the *StorageResource* class via the *hasAccessResource* relationship.

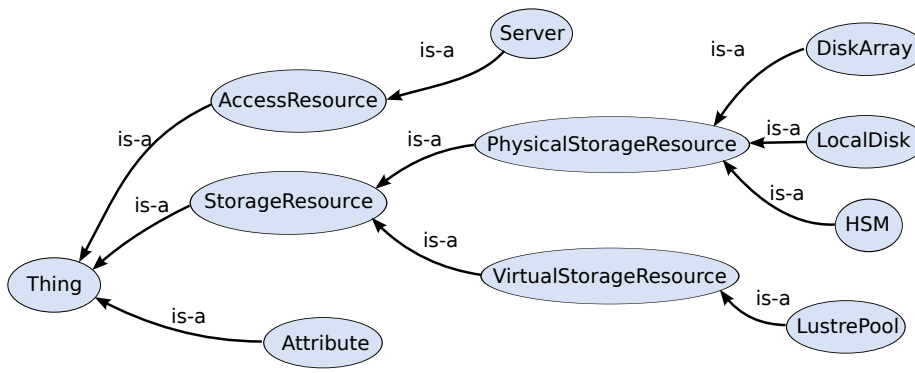


Fig. 3. An ontology describing storage resources

For testing purpose, an instance of the ontology was developed to describe distributed environment in, which the tests described below were performed. The instance ontology is used by the SES and SMED subsystems to:

- select appropriate method of measuring parameters of a resource depending on the resource type,
- find the most suitable storage node for the given non-functional requirements but only from the available ones,
- request values of quality parameters for storage resources depending on the resource type.

In Figure 4, a sequence diagram of the main QStorMan toolkit use case, i.e., finding the most suitable storage node for given requirements is depicted. The sequence diagram contains only actions, which are relevant to the semantic interaction between components. An instance of SMED in each site retrieves storage resources for the site from GOM at a start time. This is a one time operation. The main part of the use case starts when a client, e.g., one of the SES-libraries, sends a request for finding the most suitable storage node for the given non-functional requirements. The first action performed by the SES-service after retrieving the request is getting all of the available storage nodes. Then, from each SMED instance the SES-service collects current values of the given parameters. As a result, information about the most suitable storage node is returned to the SES-libraries.

4.3 Non-Functional Requirements Ontology

The second ontology used by the QStorMan toolkit defines non-functional requirements for storage resources at different levels of abstraction. The ontology is depicted in Figure 5. Non-functional requirements (represented by the *NonFunctionalRequirement* class) can be linked with a Virtual Organization, a user or an application via an instance of the *DataSLA* class. At the Virtual Organization level, the defined requirements can be treated as global settings for all users who are members of the concrete VO. Currently, only administrators of the Virtual Organization can define requirements at this level directly in the VO knowledge base. At the user level, the defined requirements regard all of the users applications. The lowest level of requirements concerns the application themselves, i.e., each application can have different requirements defined.

Both the user and the application requirement levels are supported by the *QStorMan* portlet. Basic operations such as creation, overwriting and removing are provided by clickable and intuitive Graphical User Interface in the portal (see Section 3.1). For testing purpose, an instance of the ontology was developed to describe sample requirements for test users. The instance ontology is used by the SES subsystem, i.e., *libSES* and *SES service*, to:

- get requirements of an application or virtual organization for a given user,
- find a Virtual Organization of which a given user is a member.

5 EXPERIMENTAL EVALUATION

The toolkit is developed within the PL-Grid project [28] whose one of the main goals is to support the Polish scientific community with new tools and programming frameworks that will enable scientists to run advanced scientific applications in a fast and reliable manner on Grid-based distributed infrastructure. The FiVO/QStorMan toolkit is one of the tools that will be deployed at the PL-Grid production infrastructure. Specifically, the *QStorMan portlet* is going to be embedded into the PL-Grid

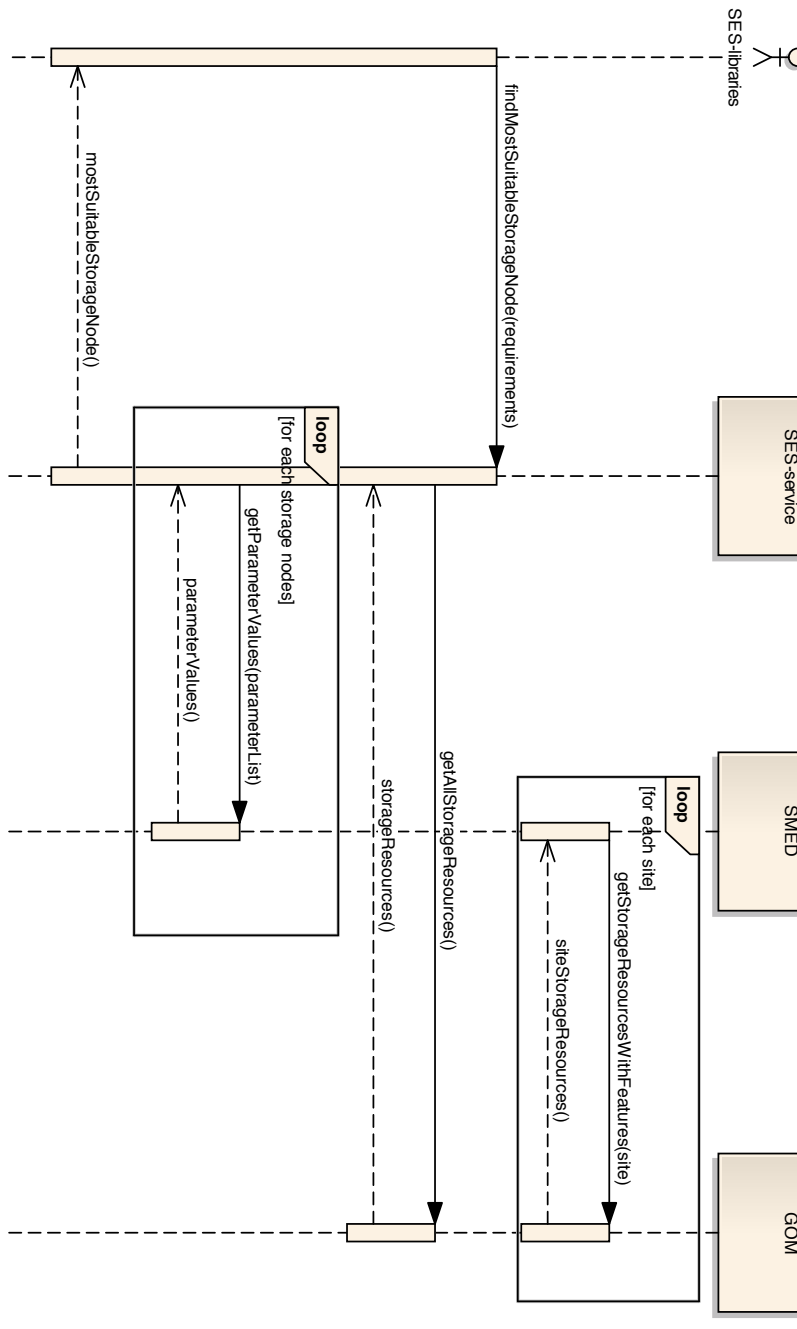


Fig. 4. A sequence diagram of the "finding storage node" use case

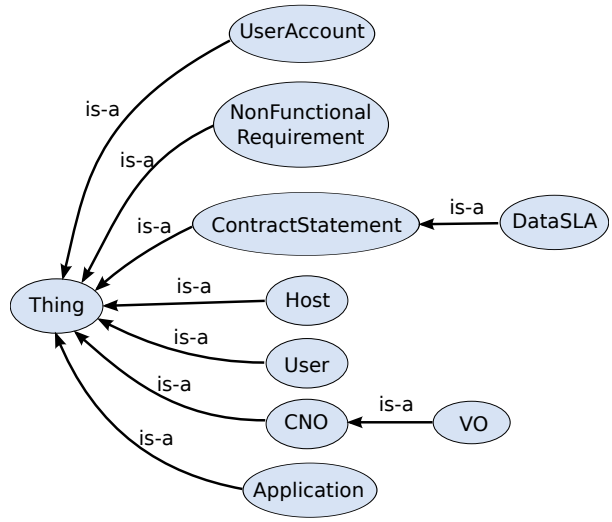


Fig. 5. An ontology describing non-functional requirements for storage resources

portal. An instance of each server-side component of QStorMan, i.e., *SES-service*, *GOM* and *SMED*, will be deployed at a dedicated server within each site of the project infrastructre.

The current version of the *QStorMan* toolkit distributes data within a distributed file system *Lustre* [29], which will be used in the PL-Grid project as the storage system for temporary data coming from applications running on the Grid.

In order to verify whether the presented approach to data management is efficient, a set of tests was performed. Testing computer systems, which operate in highly distributed and dynamic environments such as Grids is not a trivial task. There are several factors, which testers should keep in mind, e.g., network traffic or node failure. To overcome these problems, many simulators of Grid environments were developed, e.g., OptorSim [30], which allows to test dynamic replication strategies used in optimising data location within a Grid. Each simulated site contains several storage or computing elements. However, due to Grid complexity, it is hardly possible to simulate a Grid in every detail. Thus, the results obtained with such a simulator cannot lead to right conclusions.

Therefore, we decided to test the *QStorMan* toolkit with an infrastructure similar to the Grid. We accept the fact that there may occur perturbations on the test infrastructure caused by, e.g., a huge number of users who decide to stress the infrastructure by running many data-intensive applications. By monitoring the workload of the infrastructure, we were able to minimize the impact of such situations.

5.1 Testing Environment

As a testing environment, a part of the PL-Grid infrastructure was used. It consists of three nodes in three different Polish computing centers, namely ACC Cyfronet AGH in Cracow, Poznan Supercomputing and Networking Center (PSNC) and Interdisciplinary Center of Mathematical and Computational Modelling (ICM) in Warsaw. Characteristics of the nodes are as follows:

1. ACC Cyfronet AGH (Cracow)
   - Scientific Linux SL release 5.5 (Boron)
   - 2 × Intel® Xeon® CPU L5420 @ 2.50 GHz (4 cores, 1 thread per core)
   - 16 056 MB RAM
   - 12 TB storage capacity, 150 MB/s read transfer rate, 70 MB/s write transfer rate
2. PSNC (Poznan):
   - Scientific Linux CERN SLC release 5.5 (Boron)
   - Intel® Xeon® CPU 5160 @ 3.00 GHz (2 cores, 1 thread per core)
   - 1 000 MB RAM
   - 14 TB storage capacity, 55 MB/s read trasfer rate, 46 MB/s write trasfer rate
3. ICM (Warsaw):
   - CentOS release 5.5 (Final)
   - Intel® Xeon® CPU X3430 @ 2.40 GHz (4 cores, 1 thread per core)
   - 7 975 MB RAM
   - 5 TB storage capacity, 50 MB/s read trasfer rate, 27 MB/s write trasfer rate.

A map of the testing environment is depicted in Figure 6. The client machine, i.e., the machine from which the test scripts were executed, was a standard worker node located at the ACC Cyfronet AGH center. Both service: SES-service and GOM were deployed on different machines. Each storage node represents an access resource to the Lustre file system installation at each supercomputing center. In each supercomputing centre an instance of SMED subsystem was deployed. We omit the network connection details as well as details of storage resources connection within each supercomputing center to keep the map simple.

5.2 Testing Scenario

A developed testing scenario is a data-intensive application, which alternately performs computation and then write data to a storage node. This is a common behaviour when an application generates such amount of data, which has to be stored in an external memory. An example of this type of applications is particle simulation. Such a simulation is divided into a number of iterations and within each

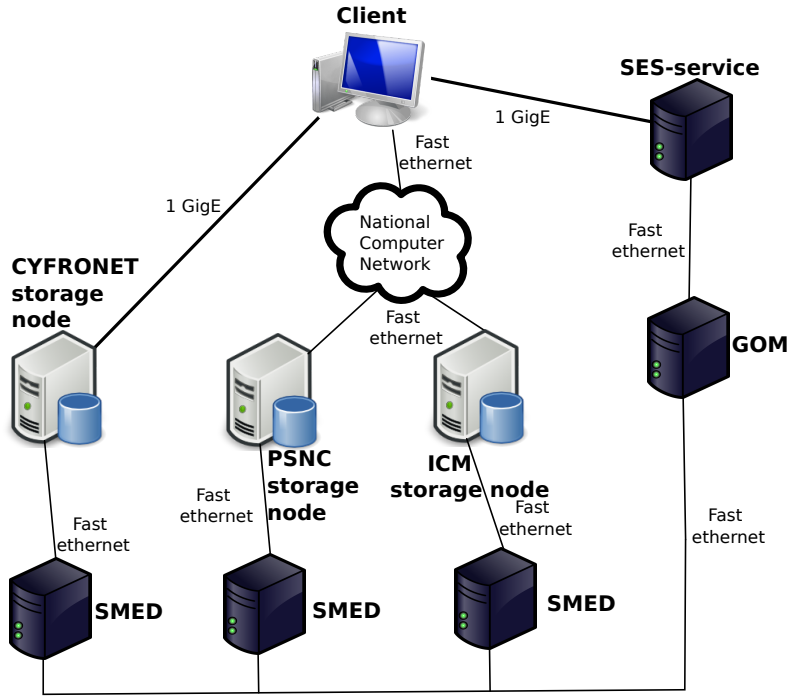


Fig. 6. A map of the QStorMan testing environment

iteration the particle's positions for the current time-step are computed and stored in an output file for later trajectory analysis. Each file represents a single iteration of such a simulation.

In the developed scenario, the *QStorMan* toolkit has been used to select a site where a job will run according to measured current write transfer rates in available sites. Thus, it simulates the use case where a user interacts with the *QStorMan* via a web portal. The scenario was parametrized with the following values:

- the number of users running in parallel: 6 (4 of which uses the QStorMan toolkit),
- a single file size output: 2 GB,
- the number of output files to write: 20, 40, 50.

5.3 Results and Discussion

The performance evaluation metric used in the presented tests is *data write time*. The metric represents total time of writing data within a single test job. In the performed tests, this is the time of writing $2 \times 20 = 40$ GB, $2 \times 40 = 80$ GB and $2 \times 50 = 100$ GB of data.

The results from the previously described testing scenario are depicted in Figure 7. The chart consists of four groups of bars that represent total data write time of a number of files from the X axis. The smaller bars mean better results. In most cases, the QStorMan users data write time was smaller from 10 % up to 40 % compared to the users who did not use QStorMan. It is worth mentioning that this scenario simulated the case where no QStorMan specific modification has to be made to the application source code.

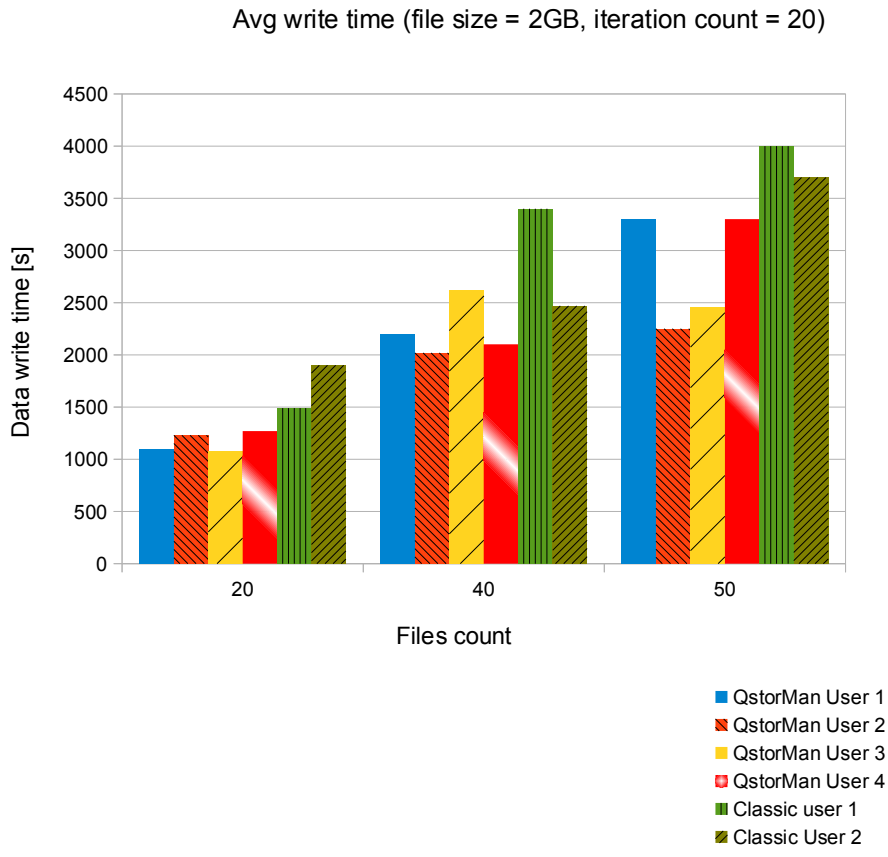


Fig. 7. Test results from data-oriented Grid job simulation

Another important parameter which was measured is an overhead generated by the *QStorMan* including the usage of semantic technologies in particular for the global execution time of users application. The overhead is presented in Table 1. As the *QStorMan* toolkit exposes its functionality via the SES-service, thus the overhead can be defined as the SES-service request response. Each request can be divided into three stages:

- gathering data from GOM about available storage resources,
- getting information from SMED about given QoS parameters of the resources,
- and finding the most suitable storage node.

Getting storage resources time [s]	Getting data from SMED time [s]	Finding storage node time [s]	Total processing request time [s]
1.007	1.123	0.003	2.134

Table 1. An overhead of the QStorMan toolkit

The results presented in Table 1 show that the SES-service processes a single request within 2.134 seconds, which is rather fast compared to data write time for a single job. Notice that 6 users run in parallel, thus there were 3 requests processed by the SES-service simultaneously. If there would be only one user, then the process time could be less than 1 second. The first stage of processing a request, i.e., getting information from GOM about available storage resources, takes about 47 % of the overall processing time while the communication with SMED takes about 52 %. This time includes the processing time by each component as well as network communication. The stage of finding the most suitable storage node is negligible due to taking less than 1 % of the overall processing time.

Exploitation of semantic technologies generates only a minor overhead compared to data write time for a job. However, it allowed us to create a storage provisioning system whose components are integrated at the semantic level. Thus it can be easily extended with e.g., new definitions of QoS parameters or types of storage devices. Also new components, e.g., an accounting tool, can be added in a loosely coupled manner if necessary.

6 CONCLUSIONS AND FUTURE WORK

In this paper we have discussed how semantic definition of requirements related to VO level data management in a data-intensive application can improve end user experience of performing high performance computing research on the Grid. The results show that allowing the VO users to define their requirements related to data management using extensible and machine processable format allows the system to optimize the data access times significantly.

An implementation of the proposed approach to the data management process – the FiVO/QStorMan toolkit – provides a few different ways to define non-functional requirements regarding storage systems. By using a programming library, the user can decide how each of the files created by an application should be managed. This approach requires modifications in the application source code. On the other hand, in our approach the user can define application-wide requirements by using a system library. In this case, no modification to the application source code is needed.

The performed tests show a possible speed up of data write time up to 40 % of a job, which is scheduled with the QStorMan toolkit comparing to a job, which

is scheduled without taking into account information about non-functional requirements of the job and the current workload of the infrastructure.

The future work will include further experiments with the proposed system in the framework of the PL-Grid project as well as application of the proposed approach to Cloud based infrastructures. Also, enhancements to the area of requirements definition are planned. By futher exploiting the semantic technologies, we plan to enable users to define their requirements at a higher, more application-specific, level of abstraction. We also consider an adoption of some fuzzy logic [31] and machine learning [32, 33] techniques for monitoring and retrieving of user's data access patterns and trends.

Acknowledgements

This research is supported partly by the European Regional Development Fund program No. POIG.02.03.00-00-007/08-00 as part of the PL-Grid Project and by Polish Ministry of Science and Higher Education Grant No. 690/N-EGI/2010/0. We thank Kornel Skałkowski and Michał Orzechowski for help in preparing and performing tests of the QStorMan toolkit. We thank Bartłomiej Burba (PSNC), Marcin Stolarek (ICM), Patryk Lasoń (Cyfronet), Marek Magryś (Cyfronet) and Łukasz Flis (Cyfronet) for help in preparing the testing environment based on the PL-Grid testing infrastructure.

REFERENCES

[1] Foster, I.—Kesselman, C.—Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations., Int. J. High Perform. Comput. Appl., Vol. 15, 2001, No. 3, pp. 200–222.

[2] Alfieri, R.—Cecchini, R.—Ciaschini, V.—dell'Agnello, L.—Frohner, Lorentey, K.—Spataro, F.: From Gridmap-File to VOMS: Managing Authorization in a Grid Environment. Future Generation Comp. Syst., Vol. 21, 2005, No. 4, pp. 549–558.

[3] Cummings, J.—Finholt, T.—Foster, I.—Kesselman, C.—Lawrence, K. A.: Beyond Being There: A Blueprint for Advancing the Design, Development, and Evaluation of Virtual Organizations. Technical report, National Science Foundation, available at `http://www.ci.uchicago.edu/events/VirtOrg2008` (retrieved March 30, 2011).

[4] Kryza, B.—Dutka, L.—Słota, R.—Kitowski, J.: Dynamic VO Establishment in Distributed Heterogeneous Business Environments. LNCS, Vol. 5545, 2009, pp. 709–718.

[5] Słota, R.: Storage QoS Provisioning for Execution Programming of Data-Intensive Applications. Scientific Programming, Vol. 20, 2012, No. 1, DOI 10.3233/SPR-2012-0339, IOS Press, pp. 6980.

[6] Haase, P.—Mathass, T.—Schmidt, M.—Eberhart, A.—Walther, U.: Semantic Technologies for Enterprise Cloud Management. http://uidops.net/wp-content/uploads/downloads/2010/08/Publications/iswc2010.pdf, as of April 2, 2011.

[7] The iRODS project website: https://www.irods.org, as of April 2, 2011.

[8] Akal, F.—Schuldt, H.—Schek, H.: Toward Replication in Grids for Digital Libraries with Freshness and Correctness Guarantees. Concurrency and Computation: Practice and Experience, Vol. 20, 2008, No. 17, pp. 1981–1993.

[9] Chen, P.—Chang, J.—Su, J.—Shieh, C.: On-Demand Data Co-Allocation with User-Level Cache for Grids. Concurrency and Computation: Practice and Experience, Vol. 22, 2010, No. 18, pp. 2488–2513.

[10] The FUSE project website: http://fuse.sourceforge.net/, as of April 2, 2011.

[11] Sashia, K.—Thanamani, A.: Dynamic Replication in a Data Grid Using a Modified BHR Region Based Algorithm. Future Generation Comp. Syst., Vol. 27, 2011, No. 2, pp. 202–210.

[12] Pérez, J.—Carballeira, F.—Carretero, J.—Caldern, A.—Fernndez, J.: Branch Replication Scheme: A New Model for Data Replication in Large Scale Data Grids. Future Generation Comp. Syst., Vol. 26, 2010, No. 1, pp. 12–20.

[13] Słota, R.—Nikolow, D.—Skitał, L.: Implementation of Replication Methods in the Grid Environment. LNCS, Vol. 3470, 2005, pp. 474–484.

[14] Słota, R.—Skitał, L.—Nikolow, D.: Algorithms for Automatic Data Replication in Grid Environment. LNCS, Vol. 3911, 2006, pp. 707–714.

[15] Myłka, A.—Myłka, A.—Kryza, B.—Kitowski, J.: Integration of Heterogeneous Data Sources Into an Ontological Knowledge Base. Computing and Informatics, Vol. 30, 2012, No. 1, pp.189–223.

[16] OWL Web Ontology Language project website: http://wiki.lustre.org/index.php/Creating_and_Managing_OST_Pools, as of April 2, 2011.

[17] Eclipse project website: http://www.eclipse.org/, Retrieved on April 4, 2011.

[18] XACML standard: http://www.oasis-open.org/committees/xacml/, as of April 2, 2011.

[19] Kryza, B.—Słota, R.—Majewska, M.—Pieczykolan, J.—Kitowski, J.: Grid Organizational Memory-Provision of a High-Level Grid Abstraction Layer Supported by Ontology Alignment. Future Generation Comp. Syst., Vol. 23, 2007, No. 3, pp. 348–358.

[20] Fielding, R. T.: Principled Design of the Modern Web Architecture. ACM Transactions on Internet Technology, Vol. 2, 2002, pp. 115–150.

[21] The Job Description Language (JDL reference website: http://www-numi.fnal.gov/offlinesoftware/srt_public_context/GridTools/docs/jobs_jdl.html, as of April 2, 2011.

[22] The Liferay portal website: http://www.liferay.com/, as of April 2, 2011.

[23] JSR-168 specification. http://jcp.org/aboutJava/communityprocess/_nal/jsr168/index.html, as of April 2, 2011.

[24] Fibinger, J.—Puzon, B.—Kryza, B.—Słota, R.—Kitowski, J.: Virtual Organization Security Layer Deployment Assistance. In: M. Bubak, M. Turala, K. Wiatr (Eds.), Proceedings of Cracow Grid Workshop – CGW '09, 2009, ACC-Cyfronet AGH Krakow 2010, pp. 88–95.

[25] The OntoStor project website: http://www.icsr.agh.edu.pl/ontostor/en.html, as of April 2, 2011.

[26] Polak, S.—Słota, R.: Organization of Quality-Oriented Data Access in Modern Distributed Environments Based on Semantic Interoperability of Services and Systems. In: Salvatore F. Pileggi and Carlos Fernandez-Llatas (Eds.), Semantic Interoperability: Issues, Solutions, Challenges, River Publishers, 2012, ISBN 978-87-92329-79-0, pp. 131–152.

[27] The Lustre pool mechanism: http://wiki.lustre.org/index.php/Creating_and_Managing_OST_Pools, as of April 2, 2011.

[28] The PL-Grid project website: http://www.plgrid.pl, as of April 2, 2011.

[29] The Lustre filesystem website: http://wiki.lustre.org, as of April 2, 2011.

[30] Bell, W.—Cameron, D.—Millar, A.—Capozza, L.—Stockinger, K.—Zini, F.: Optorsim: A Grid Simulator for Studying Dynamic Data Replication Strategies. International Journal of High Performance Computing Applications, Vol. 17, 2003, No. 4, pp. 403–416.

[31] Funika, W.—Szura, F.—Kitowski, J.: Agent-Based Monitoring Using Fuzzy Logic and Rules. Computer Science, Vol. 12, 2011, pp. 103–113, ISSN 1508-2806.

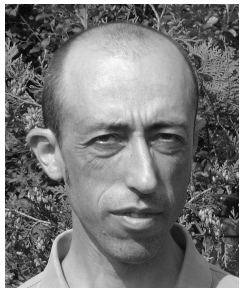
[32] Śnieżyński, B.—Dajda, J.: Comparison of Strategy Learning Methods in Farmer-Pest Problem for Various Complexity Environments Without Delays. Journal of Computational Science, 2012, ISSN 1877-7503, Available online 30 March 2012, 10.1016/j.jocs.2012.03.003.

[33] Śnieżyński, B.: Agent Strategy Generation by Rule Induction. Computing and Informatics, in press.

Renata Słota works at the Department of Computer Science of the AGH University of Science and Technology in Krakow, Poland. She obtained her Ph. D. in 1998 in computer science at the same university. She is the author or co-author of about 110 scientific papers. Her topics of interest include distributed systems, grid and cloud environments, data management and storage systems, knowledge engineering. She has been involved in many national (currently: PLGrid Plus, KMD2) and international projects, most notably in EU IST: CrossGrid, Pellucid, K-WfGrid, GREDIA, and Int.eu.grid projects. She is a Program Committee member of International Conference on Computational Science (ICCS) and reviewer of: Computing and Informatics (CAI), Future Generation Computer Systems (FGCS), and Computer Science (CSCI) journals.

Dariusz Król received his M. Sc. in Computer Science at the University of Science and Technology in Krakow, Poland, in 2009. Now he is a Ph. D. student at the same university and a scientific programmer at ACC CYFRONET AGH. He is the author or co-author of about 20 scientific papers. He has been involved in many national and international projects, funded by European Commission and EDA, e.g. ViroLab, GREDIA, PL-Grid, EDA EUSAS. He is a Program Committee member of International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING) and reviewer of the Computer Science (CSCI) journal. His topics of interest include cloud computing, storage systems, autonomic computing, large-scale web applications and high availability systems.

Bartosz Kryza is researcher and developer at the Academic Computer Center CYFRONET in Krakow. He has participated in several EU-IST projects as task or WP leader, including FP5 CrossGrid, FP5 Pellucid, FP5 MAGIC (during research internship in France), FP6 K-Wf Grid, FP6 GREDIA and FP7 PRACE. His main areas of interest are at the convergence of Grid systems and semantic technologies, SOA architectures and virtual organisations, distributed data management and P2P technologies. He is the author or co-author of about 30 research papers published in international journals or conference proceedings.



Darin Nikolow obtained his Ph. D. in Poland at the AGH University of Science and Technology in 2003. He is a lecturer at the Department of Computer Science of AGH UST with specialization in data protection technologies. His reserch interests include storage systems and distributed computing.

Jacek Kitowski (Full Professor of Computer Science) graduated in 1973 at Electrical Department of the AGH University of Science and Technology in Krakow (AGH-UST, Poland). He obtained his Ph. D. in 1978 and D. Sc. in 1991 in computer science at the same university. He is the Head of Computer Systems Group at the Department of Computer Science of the AGH University of Science and Technology in Cracow, Poland, and senior researcher at the Academic Computer Centre CYFRONET-AGH, being responsible for developing high-performance

systems and grid environments. He is the author or co-author of about 200 scientific papers. His topics of interest include large-scale computations, multiprocessor architectures, parallel/distributed computing, Grid services and Cloud computing, SOA systems, knowledge engineering and semantic technologies. He participates in program committees of many conferences, and was/is involved in many international and national projects, like EU funded Crossgrid, Pellucid, int.edu.grid, K-WfGrid, Gredia, gSLM and EDA EU-SAS. He is Polish expert (nominated by the Ministry of Science and Higher Education) in EU Program Committee e-Infrastructures (EU Unit F3 Research Infrastructures) and Director of PL-Grid Consortium coordinating the PL-Grid and PLGrid PLUS projects co-funded by the European Regional Development Fund as part of the Innovative Economy Program (National Grid Initiative, Polish NGI), closely cooperating with EGI.eu and EGI InSPIRE. He is a Member of the Interfaculty Commission of Technical Sciences of the Polish Academy of Arts and Sciences (PAU) and of the Computational Science Section of the Polish Academy of Sciences (PAN), Committee on Informatics. as well as the Editor-in-chief of Computer Science Journal (published by AGH-UST).